Questão 1

A estrutura básica de uma função segue o exemplo:    
*def multiplicar\_numeros(a, b):   
    return a\*b*

Nele , a função " multiplicar\_numeros"  é responsável por calcular o resultado da multiplicação entre dois números ,  denominados "a" e "b".

Explique para que servem os elementos básicos de uma função em Python ( como  parâmetros de entrada, retornos e definição da função). Você pode utilizar o exemplo fornecido como base.

Adicionalmente ,  comente quais vantagens  e dificuldades a criação de funções em um código em Python  pode trazer ,   com base na  sua própria experiência.

Resposta:

Dentre os elementos que constituem uma função, temos a definição da função, os parâmetros de entrada e os returns.

Utilizando-se do exemplo dado na questão, explicarei cada uma delas:

No exemplo, a definição da função está nomeada como “multiplicar\_numeros”, sempre seguindo da palavra ‘def’, obrigatória para se definir uma função em Python, a definição da função é o nome que a função terá, geralmente o nome é associado ou parecido com o que a função irá executar, no caso, a multiplicação de números. Seu nome sempre com letras minúscula e se houver separação, usar o ‘underline’.

Após a definição, abrem-se parênteses para se adicionar ou não os “parâmetros de entrada”, que são os dados de entrada, ou seja, dados que a função recebe para serem executados dentro dela. No exemplo da questão, se observa pelo (a, b), sempre separados por virgula e com um espaço.

E por fim, o “return” que em uma função vai retornar valores, sendo um ou mais valores, de qualquer tipo de dado, e é a indicação que a função acabou, o “return” finaliza a função, sai do bloco de execução da função. No exemplo dado ‘a\*b’.

As funções podem ser muito úteis se forem bem trabalhadas. Como experiência no curso, pude perceber tanto vantagens como dificuldades em relação a elas.

A mais clara vantagem é a própria definição do que uma função faz em um programa, elas são muito úteis para executar procedimentos similares por repetidas vezes, sem precisar ficar reescrevendo o código e deixando ele mais limpo, com parâmetros do PEP-8,. Assim como executar tarefas um pouco mais complexas a fim de deixar o código enxuto, como aceitarem dentro delas, estruturas de repetições como o *for* e o *while.*

Dentre as dificuldades, confesso que foi o tema abordado na matéria que mais me trouxe dor de cabeça, mesmo assistindo as aulas, explicações em vídeos, pra quem começou com programação agora, não é tão simples entender todos os passos de como por exemplo uma função é executada dentro de uma estrutura de repetição, e mesmo entendendo, não é fácil monta-la, e ainda sim, mesmo após ver montada, fiquei olhando por um bom tempo até entender como funciona, mas creio que com treino e pratica tudo fica mais fácil.

Questão 2

Você foi convidado  recentemente  para participar de um processo seletivo referente  a  uma vaga de estágio para atuar no desenvolvimento de algoritmos em Python  em  uma grande empresa. O processo seletivo é composto  por  algumas fases: prova de conhecimentos gerais ,  dinâmicas  em  grupo e, finalmente, entrevistas individuais com a equipe de  recursos humanos  e  uma avaliação  de conhecimentos específicos em Python.

Ao chegar à fase de avaliação de conhecimentos específicos, você recebe uma folha de papel contendo um conjunto de questões. Uma delas possui o seguinte texto: "Escreva um algoritmo em Python usando laços de repetição para mostrar na tela todos os números entre 0 e 1000 que sejam múltiplos de 3". Por laços de repetição entende-se um algoritmo que utilize o comando "for" ou o comando "while".

Explique para que servem os comandos do tipo "for" ou "while" em Python. Desenvolva um algoritmo em Python utilizando o comando "for" ou o comando "while" que mostre na tela todos os números entre 0 e 1000 que sejam múltiplos de 3. Você pode escolher um dos dois comandos.

Resposta:

**for** n **in** range(0, 1000, 3):  
 print(n)

Os laços ‘*for*’ e ‘*while*’ são laços de repetição, como o próprio nome os define (estruturas repetitivas), levando ao pé da letra o inglês, o laço *while* vai executar o bloco de códigos enquanto a condição for verdadeira, ou seja, ele é recomendado quando não sabemos o número exato de quantas vezes o comando deve se repetir.

Já o laço *for* permite ao programador percorrer itens dentro de uma coleção e executar um bloco de códigos para cada um deles, por isso é excelente para se trabalhar com listas. E como geralmente as listas e coleções são finitas, sempre sabemos o número de repetições do código.